Information Sciences 364–365 (2016) 146–155

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A survey of randomized algorithms for training neural networks

Le Zhang, P.N. Suganthan*

School of Electric and Electronic Engineering, NanYang Technological University, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 10 April 2015
Revised 12 November 2015
Accepted 17 January 2016
Available online 23 January 2016

Keywords:
Randomized neural networks
Recurrent neural networks
Convolutional neural networks
Deep learning

ABSTRACT

As a powerful tool for data regression and classification, neural networks have received considerable attention from researchers in fields such as machine learning, statistics, computer vision and so on. There exists a large body of research work on network training, among which most of them tune the parameters iteratively. Such methods often suffer from local minima and slow convergence. It has been shown that randomization based training methods can significantly boost the performance or efficiency of neural networks. Among these methods, most approaches use randomization either to change the data distributions, and/or to fix a part of the parameters or network configurations. This article presents a comprehensive survey of the earliest work and recent advances as well as some suggestions for future research.

1. Introduction

Inspired by biological Neural network, artificial neural network (ANN) is a family of non-parametric learning methods for estimating or approximating functions that may depend on a large number of inputs and outputs. Typically, training protocol of an ANN is based on minimizing a loss function defined on the desired output of the data and actual output of the ANN through updating the parameters. Classical approaches usually tune the parameters based on the derivatives of the loss function. However, much of the power of ANN comes from the nonlinear function in the hidden units used to model the nonlinear mapping between the input and output. Unfortunately, this kind of architecture loses the elegance of finding the global minimum solution with respect to all the parameters of the network since the loss function depends on the output of nonlinear neurons. Thus, the optimization turns out to be nonlinear least square problem which is usually solved iteratively. In this case, the error function has to be back propagated backwards to serve as a guidance for tuning the parameters [30]. Due to this, it is widely acknowledged that these training methods are very slow [38] and may not converge to a single global minimum because there exist many local minima [29,53] and also the resulting neural network is very weak in the real world noisy situations. These weaknesses of this family of methods naturally limit the applicability of gradient-based algorithms for training neural networks. Randomization based methods remedy this problem by either randomly fixing the network configurations (such as the connections) or some parts of the network parameters (while optimizing the rest by a closed form solution or an iterative procedure), or randomly corrupt the input data or the parameters during the training. Remarkable results have been achieved in various network structures, such as single hidden layer feed forward network [69],

* Corresponding author. Tel: +65 6705404; fax: +65 67933318.
E-mail address: epnsugan@ntu.edu.sg (P.N. Suganthan).

http://dx.doi.org/10.1016/j.ins.2016.01.039
0020-0255/

RBF neural networks [9], deep neural network with multiple hidden layers [31], convolutional neural network [43] and so on.

A main goal of the paper is to show a role and a place of randomized methods in optimization based neural networks' learning. In Section 2, we present some early work on this line of research on perceptron and standard feed-forward neural network with random parameters in the hidden neuron. Another piece of important work is Random Vector Functional Link Network, which is described in Section 3. Randomization based learning in RBF, recurrent neural network and deep neural network are presented in Sections 4, 5, and 6, respectively. We also offer some details on other scenarios such as evolutionary learning in Section 7. In Section 8, we point out some research gap in the literature of randomization algorithm for neural network training. Conclusions are presented in the last section.

2. Early works on perceptron and standard feed-forward neural network with randomization

The earliest attempt in this research area was the "perceptron" presented in [65] and extended in [10,66]. Generally speaking, a perceptron consists of a retina of sensor units, associator and response units. The sensor units are connected to the associator units in "random and many to many" manner. The associator units may connect to other associator units and/or response units. When a stimulus (or the input data) is presented to the sensor units, impulses are conducted from the activated sensor units to the associator units. The associator units are activated once the total arrived signals exceed a threshold. In this case, an impulse from the associator will be send to the units which are connected with it. In perceptron, the weights between the sensor units and the response units can be regarded as randomly selected from {1, 0}, while the weights between the associator units and the response units are achieved by reinforcement learning.

In [69], the authors investigated the performance of a standard feed-forward neural network (SLFN) which is demonstrated in Fig. 1. In this paper, the weights between the input layer and hidden layer are randomly generated and kept fixed. The author reported that the weights between the output layer and hidden layer are of more importance and the rests may not need to be tuned once they are properly initialized.

For a given classification problem with limited training data, there are numerous solutions with different parameter setting which is statistically acceptable. In this case, training become much easier because the learning set is only needed to make a rough selection in the parameter space. Setting the parameters in the hidden neurons randomly helps to remove the redundancy of the solution in parameter space and thus makes the solution less sensitive to the resulting parameters compared with other typical learning rule such as back-propagation. In [69], the weights in hidden neurons are set to be uniform random values in $[-1, +1]$ and they suggest to optimize this range in a more appropriate range for the specified application. An alternative choice is to set the hidden neurons to act as "correlators", which means to fix the weights in hidden neuron with a random subset of the training data.

In [69], the network's output layer weights are optimized by minimizing the following squared error:

$$\epsilon^2 = \sum_{i=1}^{N} \left(y_i - \sum_{j=0}^{k} w_j f_{ij} \right)^2 \tag{1}$$

where N means the number of data samples and k is the number of hidden neurons. f_{ij} is the activation values of the jth neuron on the ith data sample (f_{i0} is the bias). y_i is the target of the ith data sample.

Denote by

$$F_i = [f_{i0}, f_{i1}, \ldots, f_{ik}]^T \tag{2}$$

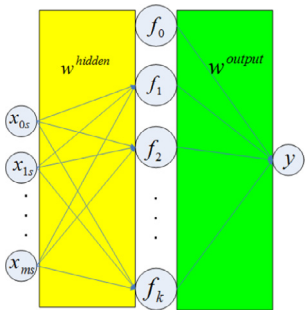

Fig. 1. The structure of SLFN in [69]. x means the input feature. The arrows within the yellow rectangle represents the random weights (w^{hidden}) which connect the input feature to the hidden neurons. Those arrows within the green rectangle are the output weights (w^{output}) which need to be optimized. x_{0s} and f_{0s} can be regarded as the bias term in the input and hidden layer. y is the desired output target. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)